WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
(deadlock or livelock) same bus same (bridge or expansion)	120

Database: US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L2

Refine Search

Recall Text | Clear

Search History

DATE: Tuesday, June 03, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L2	(deadlock or livelock) same bus same (bridge or expansion)	120	L2
L1	(deadlock or livelock) same bus same bridge	117	L1

END OF SEARCH HISTORY

WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
L2	0

Database: US Patents Full-Text Database; US Pre-Grant Publication Full-Text Database; JPO Abstracts Database; EPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search: L3

Refine Search | Recall Text | Clear

Search History

DATE: Tuesday, June 03, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L3	L2	0	L3
DB=USPT; PLUR=YES; OP=OR			
L2	(deadlock or livelock) same bus same (bridge or expansion)	120	L2
L1	(deadlock or livelock) same bus same bridge	117	L1

END OF SEARCH HISTORY

WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
((709/100 |709/208)!.CCLS. |(710/110 |710/107 |710/263 |710/41 |710/52)!.CCLS. |(711/151)!.CCLS. |(714/47)!.CCLS.)	3862

Database: US Patents Full-Text Database; US Pre-Grant Publication Full-Text Database; JPO Abstracts Database; EPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search: L4

Refine Search | Recall Text | Clear

Search History

DATE: Tuesday, June 03, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L4	((709/100 |709/208)!.CCLS. |(710/110 |710/107 |710/263 |710/41 |710/52)!.CCLS. |(711/151)!.CCLS. |(714/47)!.CCLS.)	3862	L4
DB=PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L3	L2	0	L3
DB=USPT; PLUR=YES; OP=OR			
L2	(deadlock or livelock) same bus same (bridge or expansion)	120	L2
L1	(deadlock or livelock) same bus same bridge	117	L1

END OF SEARCH HISTORY

WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
l1 and L4	37

Database: US Patents Full-Text Database; US Pre-Grant Publication Full-Text Database; JPO Abstracts Database; EPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search: L5

Refine Search | Recall Text | Clear

Search History

DATE: Tuesday, June 03, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L5	l1 and L4	37	L5
L4	((709/100 |709/208)!.CCLS. |(710/110 |710/107 |710/263 |710/41 |710/52)!.CCLS. |(711/151)!.CCLS. |(714/47)!.CCLS.)	3862	L4
DB=PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L3	L2	0	L3
DB=USPT; PLUR=YES; OP=OR			
L2	(deadlock or livelock) same bus same (bridge or expansion)	120	L2
L1	(deadlock or livelock) same bus same bridge	117	L1

END OF SEARCH HISTORY

Your search matched **[0]** of **[942182]** documents.

You may refine your search by editing the current search expression or entering a new one the text box. Then click search Again.

((deadlock or livelock) and bus)and ((bridge or expansion))

Search Again

OR

Use your browser's back button to return to your original search page.

Results:

No documents matched your query.

Your search matched **3** of **940420** documents.

A maximum of **3** results are displayed, **25** to a page, sorted by **Relevance** in **descending** order.
You may refine your search by editing the current search expression or entering a new one the text b
Then click **Search Again**.

((deadlock or livelock) and bus)

Search Again

Results:
Journal or Magazine = **JNL** Conference = **CNF** Standard = **STD**

1 **Irregular torus networks: deadlock avoidance and throughput analys**
Chen, Y.-M.; Sasaki, G.H.;
INFOCOM '93. Proceedings.Twelfth Annual Joint Conference of the IEEE Compu Communications Societies. Networking: Foundation for the Future. IEEE , 1993
Page(s): 312 -321 vol.1

[Abstract] [PDF Full-Text (732 KB)] **IEEE CNF**

2 **Cellular logic bus arbitration**
Adamides, E.D.; Iliades, P.; Argyrakis, I.; Tsalides, Ph.; Thanailakis, A.;
Computers and Digital Techniques, IEE Proceedings E [see also Computers and Techniques, IEE Proceedings-] , Volume: 140 Issue: 6 , Nov 1993
Page(s): 289 -296

[Abstract] [PDF Full-Text (420 KB)] **IEE JNL**

3 **A scalable multibus configuration for connecting transputer links**
Adda, M.;
Parallel and Distributed Systems, IEEE Transactions on , Volume: 8 Issue: 3 , M 1997
Page(s): 245 -253

[Abstract] [PDF Full-Text (192 KB)] **IEEE JNL**

A scalable multibus configuration for connecting transputer links

Adda, M.
Richmond the American Int. Univ., London;

This paper appears in: **Parallel and Distributed Systems, IEEE Transacti on**
On page(s): 245-253
Volume: 8, Issue: 3, Mar 1997
ISSN: 1045-9219
References Cited: 26
CODEN: ITDSEO
INSPEC Accession Number: 5550306

Abstract:
The paper presents the development and the performance of a novel bus bas message passing interconnection scheme which can be used to join a large number of INMOS transputers via their serial communication links. The main feature of this architecture is that it avoids the communication overhead whic occurs in systems where processing nodes relay communications to their neighbors. It also produces a flexible and scalable machine whose attractive characteristics are its simplicity and low latency for large configurations. We s that this architecture is free from deadlock, exhibits much smaller latency tha most directly connected transputer networks and has a scalable bandwidth, i contrast to other bus topologies

Index Terms:
concurrency control message passing multiprocessor interconnection networks syste buses transputers INMOS transputers bus based message passing interconnection scheme bus topologies deadlock free architecture directly connected transputer netw processing nodes scalable bandwidth scalable machine scalable multibus configurati serial communication links transputer link connections

Documents that cite this document
Select link to view other documents in the database that cite this one.

L5: Entry 10 of 37 File: USPT May 1, 2001

DOCUMENT-IDENTIFIER: US 6226704 B1
TITLE: Method and apparatus for performing bus transactions orderly and concurrently in a bus bridge

Abstract Text (1):
The present invention provides a method and apparatus for performing bus transactions orderly and concurrently in a bus bridge. To meet the ordering rules, the invention adopts a HOLD/HLDA handshaking mechanism to control the flow of transactions in the bus bridge. When both HOLD and HLDA signals are asserted, the bus bridge holds the transaction processed in one direction and then the bus bridge is ready to process the transaction from another direction. That is, the bus bridge first controls the transaction flowing in one direction whenever there is request coming from another direction, wherein the HOLD signal is asserted simultaneously. Upon receipt of the HLDA signal indicating that the transaction flow has been completely held in one direction, the bus bridge allows the transaction to flow from another direction by granting the request agent bus ownership. The present invention also provides a method to avoid deadlock. The bus bridge retries transactions stalling the bus in two cases. First, the bus bridge retries non-postable transactions until the posted transactions in the posting buffers on the same side are completed at the destination. Second, the bus bridge retries postable transactions until the posting buffers on the same side have sufficient spaces to accept transactions.

Brief Summary Text (9):
As far as a bus bridge is concerned, it is responsible for maintaining transaction ordering and avoiding deadlock. Maintaining transaction ordering is mainly to have a consistent view of data in a system with write posting being allowed. Since the memory write completes at the source before it actually completes at the intended destination, the master issuing this write transaction also sets the flag to indicate that the data is now valid for other masters to use. So it right be possible that a master, regardless of which bus the master resides, reads the flag and confines the data before it is actually written to the destination. The data coherence of the system is destroyed after a master reads the stale data. As for the coherency concern, it is required to obey the ordering rule that the posted data must be written to the destination before other masters observe the valid flag and read the data. In other words, the posting buffers within the bus bridge must be flushed before the bus bridge performs a read transaction.

Brief Summary Text (10):
In addition to maintaining transaction ordering, the bus bridge should also avoid deadlock situations within the bridge. Deadlock situations typically require at least a temporary suspension of system operation, if not an entire system reset. A deadlock situation arises, for example, if the bridge contains two requests, one targeting an agent on the first bus and the second targeting an agent on the second bus, and neither request can be executed until the other is satisfied. Therefore, the deadlock prevents the bridge from operating properly.

Brief Summary Text (11):
In the existing X86 PC systems, a deadlock may occur if an I/O device makes acceptance of a memory write transaction as a target contingent on the prior completion of a memory writ e transaction as a master . If the prior write transaction initiated by the I/O master is destined for L2 cache 12/system memory

14, two deadlock situations may present in the system. One, the bus bridge 13 does not allow the I/O master to access L2 cache 12/system memory 14 by withholding the I/O bus 18 ownership from the requesting I/O master due to the posting buffers haven't been flushed. And then the posted transactions from the processor bus 17 to I/O bus 18 can not be executed at the destination due to the I/O device refuses to be a target while it can not perform a memory write first. The other, the bus bridge 13 can not hold the processor bus 17 because the processor bus 17 is stalled. There are two possible causes. First, the current outstanding transaction on processor bus 17 destined for I/O bus 18 is non-postable and waits for response until the transaction is completed on I/O bus 18. According to the ordering rule mentioned above, this non-posted transaction can not be executed on I/O bus 18 unless the posting buffers are flushed. Therefore, if some write transactions originating prior to the non-posted transaction on processor bus 17 have been posted in the posting buffers, the non-posted transaction queues up after them and stalls the processor bus 17. Second, the current outstanding transaction on processor bus 17 destined for I/O bus 18 is postable. And the processor bus 17 is stalled when the posting buffers are full such that the transaction can not be posted. As a result, the bus bridge 13 can not hold the processor bus 17 and execute the memory write requested by the I/O master, then the posted transactions in the posting buffers can not be executed on I/O bus 18 due to the I/O device 18 refuses to be a target.

Brief Summary Text (12):
In the prior X86 PC system, the I/O bus masters are allowed to access L2 cache 12/system memory 14 only after the bus bridge 13 holds and takes over the processor bus 17 while the posting buffers are flushed. During the period of being held, the processor 11 suspends the advanced outstanding transactions temporarily. Therefore, the bus bridge 13 can not promote system performance by having the write transactions moving in the opposite directions through the bridge executed concurrently. Furthermore, some deadlock situations may occur when the specific I/O devices 16 are resided on I/O bus 18, wherein the I/O devices 16 require making acceptance of a memory write transaction as a target contingent on the prior completion of a memory write transaction as a master.

Brief Summary Text (13):
Therefore, there is a need to provide a system which prevents the occurrence of deadlocks within the bus bridge, while at the same time performing bus transactions orderly and concurrently.

Brief Summary Text (16):
It is another object of the present invention to provide a deadlock-free technique for supporting concurrent processing in a bus bridge compliant with highly pipelined processor bus, such as the Pentium II processor bus.

Detailed Description Text (13):
As to the deadlock avoidance, two deadlock situations may occur in the conventional bus bridge of the X86 PC systems. The deadlock may occur if an I/O device 216 makes acceptance of a memory write transaction as a target contingent on the prior completion of a memory write transaction as a master. In the first case, the bus bridge 213 does not allow the I/O device 216 to access system memory 214 by withholding the I/O bus 218 ownership from the requesting I/O device 216 due to the HSB 202 have not been flushed. And then the posted transactions from the processor bus 217 to the I/O bus 218 can not be executed at the destination due to the I/O device 216 refuses to be a target while it can not perform a memory write transaction first. This deadlock is solved because the HSI 201 asserts a HLDA signal after the nonposted transactions in the HSB 202 are all completed on the I/O bus 218. That is, the write transactions from the I/O device 216 to the system memory 214 are allowed to be executed while the MCU 203 attempts to execute the posted transactions in the HSB 202 on the I/O bus 218.

Detailed Description Text (14):
In the second case, the processor bus 217 is stalled by the current outstanding transaction destined for the I/O bus 218 for two possible reasons. First, the outstanding transaction on the processor bus 217 is non-postable. The processor 211 is stalled to wait for the response until the transaction is completed on the I/O bus 218. But according to the ordering, this non-posted transaction can not be

executed on the I/O bus 218 if some write transactions originated prior to the non-posted transaction on the processor bus 217 have been posted in the HSB 202. Second, the outstanding transaction on the processor bus 217 is postable. The processor bus 217 is stalled when the HSB 202 is full such that the transaction can not be posted. Consequently, the bus bridge 213 can not hold the processor bus 217 to execute the memory write transaction requested by the I/O device 216, then the posted transactions in the HSB 202 can not be executed on the I/O bus 218 due to the I/O device 216 refuses to be a target. In order to avoid this deadlock situation, the HSI 201 does not commit the non-postable transactions to the HSB 202 temporarily if there is any posted transactions in the HSB 202, which have not been completed on the I/O bus 218. Meanwhile, the HSI 201 does not commit the postable transactions to the HSB 202 temporarily if the HSB 202 is not available to buffer the posted data. Such approaches of retrying the transactions from the processor bus 217 to the I/O bus 218 will prevent the processor bus 217 from being stalled under specific deadlock situations. Therefore, the deadlock avoidance is achieved through the approaches.

Detailed Description Text (16):
In summary, the bus bridge 213 of present invention preconsiders the ordering requirements and deadlock avoidance at the beginning phase when the I/O devices 216 on the I/O bus 218 arbitrate for the bus ownership. In addition, the bus bridge of the present invention adopts the mechanism mainly controlled by HOLD, HLDA and RTY_HOLD signals to perform the bus transaction orderly and concurrently. Therefore, the performance of the personal computer is enhanced. Furthermore, the bus bridge of the present invention utilizes some approaches to prevent the processor bus 217 from being stalled to avoid the deadlock which may occur in prior systems.

Current US Cross Reference Classification (2):
709/100

Current US Cross Reference Classification (3):
710/107

L5: Entry 21 of 37 File: USPT Sep 7, 1999

DOCUMENT-IDENTIFIER: US 5949981 A
TITLE: Deadlock avoidance in a bridge between a split transaction bus and a single envelope bus

Brief Summary Text (14):
In a traditional pipelined implementation, data bus tenures are kept in strict order with respect to address tenures. However, external hardware can further decouple the address and data buses, allowing the data tenures to occur out of order with respect to the address tenures. Second-generation PowerPC computers include computers whose architecture was especially designed for high performance and that incorporated such hardware. This architecture supports true split-bus operation with ordered slaves and ordered masters. "Ordered" means each master and each slave has its own independent FIFO structure supporting "ordered" service to transactions posted to it. If a slave receives three transactions A, B, and C, then it will respond to A first, B second, and C third. If a master performs transactions D, E, and F, then it expects servicing of those transactions in the order of D first, E second, and F third. There can be up to some number of outstanding master/slave pair transactions in the architecture at one time. In an exemplary embodiment, there can be up to three outstanding master/slave pair transactions in the architecture at one time. As a result, in the foregoing architecture, a bridge, such as a video bridge, may concurrently have one outstanding slave transaction to it and one outstanding master transaction from it. Although ordered masters and slaves, as opposed to unordered masters and slaves, provide an overall simplification to system architecture, they can lead to deadlocks when there are conflicting completion dependencies.

Brief Summary Text (15):
Deadlock occurs in a system when one resource cannot complete an access to another resource, and the access blocks other resources from performing transactions on the bus. In the case of a bridge between a split bus and a single-envelope bus (such as the ARBus and the PCI bus), since the different buses are referenced to different clock signals of different frequencies and phases, transactions that pass from one side of the bus bridge to the other must pass an asynchronous boundary. An attempt to avoid deadlock by passing information across this boundary will lead to some uncertainty, such that not all deadlocks can be reliably avoided using such a technique.

Detailed Description Text (29):
Referring to FIG. 7, a control portion, or control block, 710 of the video bridge 700 will be described. A datapath portion of the video bridge 700 stores or forwards data within a video and graphics subsystem of the computer system of FIG. 2. The datapath portion of the video bridge 700 is not particularly germane to the present invention. Further details concerning the arbiter 600 and split-bus operation may be found in copending U.S. Pat. No. 5,592,631 entitled "Bus Transaction Reordering Using Side-Band Information Signals, filed concurrently herewith (Attorney's Docket No. P1605/172), incorporated herein by reference. Additional details concerning the expansion bus bridge 219 may be found in copending U.S. application Ser. No. 08/432,622 entitled "Deadlock Avoidance in a Split-Bus Computer System," filed concurrently herewith (Attorney's Docket No. P1473/134), also incorporated herein by reference.

Detailed Description Text (33):
As explained previously, systems are most prone to deadlocks and live-locks when

there is a bridge in the system. In particular, in the system of FIG. 2, a Read/Read deadlock can occur when a bus master (i.e. CPU 203) on the ARBus 204 first attempts a read access (RD access 1) from the PCI bus 736 via the video bridge 700, and second enqueues a write access (WR access 2) to a second slave such that as the memory system 209, followed by another bus master (not shown) on the PCI bus 736 beginning a read access (RD access 3) from the same second slave (i.e. memory system 209) on the ARBus 204. This situation is illustrated in FIG. 6.

Detailed Description Text (35):
The video bridge 700 operates to avoid the Read/Read deadlock condition by disallowing the second deadlocking (RD access 3) read transaction. While there is an outstanding read transaction in either the master or slave portions of the split-response bus interface, the other portion will refuse to accept, or retry, another potentially deadlocking read transaction.

Detailed Description Text (37):
FIG. 6 illustrates how a Read/Read deadlock condition as described above is avoided. Specifically, FIG. 6 illustrates the enqueuing of accesses within the ARBus Slave 715 and ARBus Master 711 which each reside within the PCI bridge control circuitry 710. FIG. 6 further shows the assertion of signal 731 in accordance with the method and system of the present invention. Signal 731 indicates that an ARBus to PCI bus read is in progress and causes the ARBus Master 711 to refuse to accept another potentially deadlocking read transaction, i.e. RD access 3.

Current US Cross Reference Classification (1):
710/107

CLAIMS:

1. A method of avoiding system deadlock in a computer system having a split-transaction bus and a single-envelope bus bridged by a bus bridge, at least one master device and at least one slave device being connected to said split-transaction bus, said bus bridge having a first master control circuit and a first slave control circuit for said split-transaction bus and coupled to said split transaction bus, said method comprising the steps of:

enqueuing in said first master control circuit read transactions from said single-envelope bus to said split-transaction bus;

enqueuing in said first slave control circuit read transactions from said split-transaction bus to said single-envelope bus;

in a condition when an outstanding read transaction is enqueued in one of said first master control circuit and said first slave control circuit, signaling said outstanding read transaction condition from one to the other of said first master control circuit and said first slave control circuit; and

said other of said first master control circuit and said first slave control circuit, accepting some transactions but refusing to accept, from a respective one of said single-envelope bus and said split-transaction bus, a subsequent read transaction.